A STUDY OF LIGHTWEIGHT CRYPTOGRAPHY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ZELİHA ÇAMUR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

JULY 2020

Approval of the thesis:

A STUDY OF LIGHTWEIGHT CRYPTOGRAPHY

submitted by **ZELİHA ÇAMUR** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. Ömür Uğur _____
Director, Graduate School of **Applied Mathematics**

Prof. Dr. Ferruh Özbudak _____
Head of Department, **Cryptography**

Assoc. Prof. Dr. Ali Doğanaksoy _____
Supervisor, **Mathematics, METU**

Assoc. Prof. Dr. Fatih Sulak _____
Co-supervisor, **Mathematics, Atılım Uni.**

Examining Committee Members:

Prof. Dr. Ferruh Özbudak
Mathematics, METU _____

Assoc. Prof. Dr. Ali Doğanaksoy
Mathematics, METU _____

Assoc. Prof. Dr. Murat Cenk
Cryptography, METU _____

Assoc. Prof. Dr. Zülfükar Saygı
Mathematics, TOBB ETÜ _____

Assoc. Prof. Dr. Oğuz Yayla
Mathematics, Hacettepe University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ZELİHA ÇAMUR

Signature :

ABSTRACT

A STUDY OF LIGHTWEIGHT CRYPTOGRAPHY

Çamur, Zeliha
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Ali Doğanaksoy
Co-Supervisor : Assoc. Prof. Dr. Fatih Sulak

July 2020, 43 pages

Technology is evolving rapidly and with technology, the internet is also changing. People used to use internet to connect to each other. But with the changes in recent years, the internet is starting to be used more to connect devices to each other. These devices can range from powerful computing devices, such as desktop computers and tablets, to resource contrained devices, such as RFID tags and sensor networks. When it comes to these constrained devices, conventional cryptographic algorithms fail to provide necessary security and performance. Therefore, specific algorithms, designed with the limitations of constrained devices in mind, called lightweight algorithms are needed. For this reason, NIST has started a Lightweight Cryptography Project to standardize lightweight algorithms. In this thesis, we first explain what lightweight cryptography is, along with its target devices and performance metrics and give two examples of lightweight algorithms, block cipher PRESENT and stream cipher TRIVIUM . After that, we summarize the Lightweight Cryptography Project and give its timeline. Lastly, we talk about ASCON , one of the second round algorithms of the Lightweight Cryptography Project.

Keywords: Lightweight Cryptography, PRESENT, TRIVIUM, ASCON

ÖZ

HAFİF SİKLET KRİPTOGRAFİ ÜZERİNE BİR ÇALIŞMA

Çamur, Zeliha
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Ali Doğanaksoy
Ortak Tez Yöneticisi : Doç. Dr. Fatih Sulak

Temmuz 2020, 43 sayfa

Günümüzde teknolojik gelişmeler çok hızlı bir şekilde olmaktadır. Bu teknolojik gelişmeler ile birlikte internet de değişmektedir. Önceden insanlar interneti birbirlerine bağlanmak için kullanırken, son yıllarda, internet daha çok cihazları birbirine bağlamak amacıyla kullanılmaktadır. Bu cihazlar, masaüstü bilgisayarları veya tabletler gibi güçlü bilgisayarlar olabildiği gibi, RFID etiketleri veya sensör ağları gibi kaynak kısıtlı cihazlar da olabilmektedir. Bu kısıtlı cihazlar söz konusu olduğunda, günümüzde kullanılmakta olan kriptografik algoritmalar, güvenlik ve performans açısından yetersiz kalabilmektedir. Bu nedenle, kısıtlı cihazların limitleri göz önünde bulundurularak tasarlanan özel algoritmalara, hafif siklet algoritmalara, ihtiyaç vardır. Bu ihtiyaçtan dolayı, hafif siklet algoritmaları standartlaştırmak için, NIST bir proje başlatmıştır. Bu tezde, öncelikle hafif siklet algoritmaları açıklayıp, bu algoritmalara iki tane örnek veriyoruz. Ardından NIST'in başlatmış olduğu projeyi özetliyor, son olarak ise bu projenin ikinci turuna kalmış bir algoritma olan ASCON'dan bahsederek tezi tamamlıyoruz.

Anahtar Kelimeler: Hafif Siklet Kriptografi, PRESENT, TRIVIUM, ASCON

ACKNOWLEDGMENTS

I would like to express my greatest thanks to my supervisor Assoc. Prof. Dr. Ali Doğanaksoy and my co-supervisor Assoc. Prof. Dr. Fatih Sulak, for their encouragements, patience, invaluable advices, and immense guidance throughout. Without their help, this thesis could not have been completed.

I would also like to thank my friends, for always being by my side and listening to me whenever I was feeling down, and helping me get through everything these past few years.

Last, but definitely not least, my special thanks to my great family, for always having my back, for their continued support through thick and thin, and for never letting me quit. I would not be where I am, if it weren't for them.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Cryptography is the science of sending a message to another person or party in a way that only the recipient can read the message. The communication channel is always assumed to be insecure, because an adversary can interrupt the message transmission and capture the message. To prevent the adversary from learning the contents of the message, the sender, usually referred to as Alice, encrypts the message, called plaintext, and sends the encrypted message, called ciphertext, to the other party, usually referred to as Bob. The encryption is done with a secret key that the two communicating parties had securely decided beforehand. Since the encryption and decryption keys are kept secret, only the people with the secret key can encrypt or decrypt the message. As long as the keys remain secret an adversary cannot understand the message, or decipher it. Thus the two parties, Alice and Bob, can comminucate freely.

An adversary trying to interrupt two communicating parties may have one of the four main goals:

1. Read the secret message.
2. Find the secret key, so that they can read all messages encrypted with that key.
3. Modify the message sent by Alice and go unnoticed by both parties.
4. Act like Alice and send a message to Bob, to make Bob think he is communicating with Alice when in reality he is communicating with the adversary. [27]

In order to prevent an adversary from reaching his goals, some security measures are

applied to cryptosystems, namely confidentiality, data integrity, authentication, and nonrepudiation.

1. Confidentiality means the transmitted message or information is kept secret, and only the authorized parties have the means to decipher the information.
2. Data integrity makes sure that the messages are not being modified. This stops the adversary from reaching their third goal.
3. Authentication helps Bob to correctly identify the sender as Alice, thus stopping the adversary from posing as Alice.
4. Nonrepudiation prevents Alice from denying she sent the message. [22]

Cryptographic algorithms are gathered under two main branches; symmetric algorithms and asymmetric algorithms. In symmetric algorithms both Alice and Bob have the same key. Since the communication channel is insecure, this key must be previously decided on through secure ways. The encryption and decryption keys are either the same, or very similar that the decryption key can easily be derived from the encryption key. But sometimes Alice and Bob cannot agree on a key beforehand. They could be very far away from each other and cannot get together to determine a secret key, and there may not be a secure way for Alice to send Bob the secret key. She cannot just send Bob a secret key through any open channel, because an adversary can interrupt the channel and get their hands on the key. Thus making the key useless. To get around this problem asymmetric algorithms, usually called public key algorithms, are used. In public key algorithms each party has their key pairs, one public and one private key. As can be understood from their names, private keys are kept secret, and public keys can be known by everyone. The public key is computed from the private key in a way that finding the private key from the public key is infeasible. Alice encrypts the message she wants to send using Bob's public key. The message can only be decrypted with the corresponding private key, which only Bob has. Therefore Alice can send a secret message even though they are far away and cannot decide on a common key together.

In order to understand how public key algorithms work we can imagine a box. For Alice to send Bob a secret message, first Bob sends Alice a box with an open padlock,

for which he has the key. Alice then can put her message in the box, and lock it with the padlock. When Bob receives the box he simply unlocks the padlock and reads the message [27]. Of course there are still some security concerns, for example an adversary can intercept the box and replace the padlock with their own lock, or put their own message in the box and act like Alice. To achieve authentication and to prevent these problems, cryptographers have developed some procedures.

1.1 Symmetric Ciphers

If we examine the symmetric ciphers in detail, we can see that symmetric ciphers can be divided into two categories; stream ciphers and block ciphers.

Stream ciphers use a keystream, obtained from the original key, and encrypts the plaintext bit by bit. Encryption is usually done by combining the plaintext bits with the corresponding keystream bits with an XOR operation. In some cases stream ciphers have some advantages over block ciphers, because there is no error propagation [16]. It means that an error made in one bit of ciphertext during transmisson only affects the decyrption of that bit and doesn't affect other bits.

Block ciphers, on the other hand, take the plaintext bits in blocks. Each block is encrypted with the same encryption function and the ciphertext blocks are produced. When the length of the plaintext is not a multiple of the block size some padding is applied to the plaintext. This padding is usually done by adding a 1 bit followed by necessary amount of 0 bits. Because the encryption function doesn't change from one block to another, same blocks of plaintext are encrypted to same blocks of ciphertext. When an adversary captures the ciphertext, they can accurately guess some information about the plaintext by using this property. In order to stop any information leakage, some modes of operation are used.

1.1.1 Modes of Operation

There are several modes of operation that can be used when encrypting a plaintext with a block cipher. NIST recommends the usage of 5 modes of operation in 2001 in

800-38A document [11]:

- Electronic Codebook (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR)

In ECB mode, each block is encrypted and decrypted independently from each other. Because the encryption function does not change, identical blocks of plaintext are encypted to identical blocks of ciphertext.

In CBC mode, the ciphertext of one block is XORed with the plaintext of the next block before the encryption. For the first plaintext block an Initialization Vector IV is used.

In CFB mode, ciphertext blocks are encrypted with the encryption function instead of the plaintext blocks. Plaintext blocks are XORed with the results of encryption function to obtain the ciphertext blocks. For the first block an IV is used.

In OFB mode, IV is repeatedly encrypted with the encryption function and the results are XORed with the plaintext blocks to obtain ciphertext blocks.

In CTR mode, a nonce and counter is encrypted and the result is XORed with the plaintext block. The counter is increased each time.

All of these modes while having different advantages also have some disadvantages. For example, some of them have parallelizable encryption and decryption but others don't. The decision of which modes of operation is to be used should be made based on the desired security and performance levels.

1.2 Asymmetric Ciphers

While modern symmetric ciphers such as AES are very secure, they have some drawbacks in practicality, namely key distribution problem, and the number of keys [22].

The key distribution problem occurs when Alice and Bob want to determine a secret key. This would be easy if they can come together and decide, but if they have no means to decide on a key in person, they have to decide on the key through a secure channel. Since the communication channel is always assumed to be insecure, becuase it can be easily hacked, this poses a problem. Even if they can somehow solve this problem, they would be facing another problem, the number of keys. If there are n users in a network, and all of the users want to communicate with each other secretly, the number of encryption keys needed would be $\frac{n \cdot (n-1)}{2}$, and each user would have $n-1$ key pairs they need to know and keep secret. This becomes exponentially infeasible as the number of people increase. The usage of asymmetric ciphers eliminate these problems. Since every user has a pair of keys, and anything encrypted with a specific public key can only be decrypted with the corresponding private key, Alice and Bob doesn't need to agree on a secret key together beforehand. In addition, nobody would need to store $n - 1$ key pairs, they only need to store their own private and public keys, and the number of key pairs needed in the network would be reduced to n [22].

Cryptographic protocols can be considered as a third main branch of cryptography, and one of the most important primitives they use is called a hash function. Therefore it would be useful to go over the definition of hash functions.

1.3 Hash Functions

A cryptographic hash function is a one-way function used to compute digital signatures and provide data integrity. Hash functions take a binary string of arbitrary length, a message, and return the hash value, a binary string of fixed length, called message digest. This message digest can act as a fingerprint for the data, and provide data integrity [26].

For security purposes, a hash function should provide resistance to some problems,

namely preimage, second preimage, and collision. For preimage resistance, given a message digest it should be infeasible to find the message that produced that message digest. For second preimage resistance, given a message digest, it should be infeasible to find a message other than the original, that also gives the same message digest. Lastly, for collision resistance, it should be infeasible to find two different messages that give the same message digest [26].

Hash functions are mainly used for digital signature schemes and message authentication codes.

1.3.1 Digital Signatures

Digital signatures do on the internet, what handwritten signatures do on paper [22]. When Alice sends Bob a message, along with its signature, Bob checks the correctness of the signature and see if the message is really from Alice. The way the digital signature schemes work is by public-key cryptography. When Alice has a message she wants to send, she encrypts the message, or signs it, with her private key. Then sends this signature along with the message. Bob can verify the signature using Alice's public key. If the message is too long however, it becomes difficult to just sign the message this way. Instead hash functions are used to simplify the signing process. A digital signature scheme by using a hash function simply entails taking the hash value of the message and signing the message digest. Therefore when Bob receives the message and the signature, he first computes the message digest and then verifies the signature. If the message has been modified by an adversary the signature won't match and Bob will know the message he received is not authentic, thus digital signatures provide data integrity. Also, because private and public keys are used, an adversary cannot pose as Alice and Alice cannot deny a message she sent, providing authentication and nonrepudiation to the system.

1.3.2 Message Authentication Codes(MACs)

Hash functions don't need a key to compute the message digest. A Message Authentication Code(MAC) on the other hand, while using a hash function, needs a key.

Therefore it can also be called a keyed hash function. Like digital signatures, a MAC also computes a signature, or a tag, and appends it to the end of the message. Unlike a digital signature however, it doesn't take the hash value of the message, instead the MAC is computed as a function of the message and a symmetric key k [22]. The result of the function is taken as the Tag, T. Because this is a symmetric system, Bob also has the key. Therefore when Alice sends the message and the tag to Bob, Bob simply computes a tag from the message and compares it with the tag he received. If the message has been modified by an adversary, the tags won't match and Bob will know that the message has been modified. Thus MACs also provide data integrity, and if Alice and Bob does not suspect that the secret key has been compromised, they can safely assume there is authentication, since only Alice and Bob has the secret key. However, because both Alice and Bob has the key, Alice can deny a message she sent, saying Bob must have forged the message himself, thus the system lacks nonrepudiation.

In this thesis, we first explain lightweight cryptography, along with its target devices and performance metrics. After that we give two examples of lightweight algorithms, block cipher PRESENT and stream cipher TRIVIUM. These ciphers were chosen because they are good representations of lightweight block ciphers and lightweight stream ciphers respectively. Later, we summarize the Lightweight Cryptography Project and give its timeline. Lastly, we talk about ASCON, one of the round 2 candidates of the project. ASCON was chosen because it is one of the candidates that shows promising performance, especially on hardware.

CHAPTER 2

LIGHTWEIGHT CRYPTOGRAPHY

According to the Law of Accelerating Returns, techological changes happen exponentially [13]. When mobile phones were first introduced, they were too big to fit into our pockets. Their only function was allowing its users to talk to each other, but they were not in widespread use by the public. Even the first smartphones were quite bulky. But nowadays smartphones are much more advanced. In about 40 years, not even half a century, the mobile phones went from being a literal brick to being a small computer that can fit in a person's pocket. With this rapid evolving of technology, the internet also evolves. The internet has been used to connect people ever since the World Wide Web went live. But with the recent technological changes, internet is starting to be used more to connect devices to each other instead of people. The term the Internet of Things (IoT) is used to describe these connections. While some of these devices are powerful computing devices such as desktop computers or tablets, there are also small computing devices or resource constrained devices in the system, such as Radio-Frequency IDentification (RFID) tags, sensor nodes and smart cards. Current cryptographic algorithms work well on powerful computing devices, but it may be hard to implement them on constrained devices. When they are implemented the performance may not be up to par, or the security may not be enough. Moreover, some of these devices may run on batteries, and once deployed, battery change might not be possible. If the power consumption of the algorithm in the device is too high, the life-span of the device would be too short. Therefore specific algorithms designed for these devices are needed. The cyrptographic algorithms designed with these limitations in mind are called lightweight algorithms and the study of designing these algorithms is called lightweight cryptography.

In [15], NIST gives an overview of lightweight cryptography, and summarizes their findings under some main titles such as target devices and performance metrics.

2.1 Target Devices

Target devices of lightweight cryptography covers a wide range of devices from desktop computers and tablets to RFID tags and sensor networks. Devices like desktop computers have a high computing power and enough area to run conventional cryptographic algorithms with no problem. Therefore they don't need other specific algorithms. Devices like RFID tags and sensor networks on the other hand, have very limited computing power and area, that they are called resource constrained devices, or just constrained devices for short. In these constrained devices, it is almost impossible to implement conventional cryptographic algorithms with the desired security and performance level. For example, in microcontrollers, the instruction sets usually consist of simple instructions. This would mean that, in order to implement a conventional algorithm, the microcontroller needs to use a lot of cycles. However, this would increase the time and decrease the performance [15]. Also, some constrained devices may work on battery power. The power consumption of conventional algorithms on constrained devices would be high. When it is easy to change the battery, this may not be a problem, but when the battery change is not possible, power consumption needs to be low. Thus these devices would need specifically designed lightweight algorithms. However, these lightweight algorithms cannot be designed with just these constrained devices in mind. Since the IoT connects these devices with the devices with high computing power, and all of these devices are supposed to interoperate, it should be possible to implement the algorithms in both kinds of devices. "It is not just the limitation of a particular device that drives the need for lightweight cryptography, but also the other devices in the application that it directly interacts with [15]."

2.2 Performance Metrics

Performance metrics can be analyzed in two categories, hardware and software metrics.

2.2.1 Hardware Metrics

When a cryptographic algorithm is implemented, there are four metrics to describe its performance on hardware level; gate area, throughput, latency, and power consumption [5].

- Gate area describes the physical area needed to implement the algorithm on hardware. It is measured in Gate Equivalents(GE), or logic blocks. Since constrained devices have limited space, the lower the gate area the better.
- Throughput describes the amount of processed bits in a time unit.
- Latency describes the amount of time passed from the first initialization of the application until the obtainment of the result.
- Power consumption describes how much power is needed in order to implement the algorithm.

2.2.2 Software Metrics

In sofware case, three metrics can be used to describe an algorithm's performance; RAM consumption, code size, and throughput [5].

- RAM consumption describes the amount of memory required to implement the algorithm.
- Code size describes how much memory the code takes up.
- Throughput once again, describes the amount of processed bits in a time unit.

Out of these metrics throughput is usually measured in bits or bytes per second, latency in seconds, power consumption in watts, and RAM consumption and code size in bytes [5]. Generally a high throughput is desired while all the others should be as low as possible. However, in real life applications, there is a threeway trade-off between security, performance, and area. Therefore it is not always possible to have a high throughput while keeping all the others low. Having a desired security level

while keeping the area small would lead to high power consumption and slow implementation. Placing the focus on security and performance optimization would require a large area. Having an ideal performance and a small area would cause security leakage. Therefore, in order to design a lightweight algorithm, the optimal trade-off should be considered.

CHAPTER 3

BLOCK CIPHER PRESENT

With the deployment of small computing devices, such as RFID tags, conventional algorithms like AES has become insufficient. Because while they are secure against known cryptanalysis attacks, they also require a large area to implement, an area which can not be provided by small devices. Lightweight cryptographic algorithms have potential to solve this problem. PRESENT is a very good example of lightweight algorithms. It has been designed with the goal of hardware optimization. Therefore the power consumption and area have been given the utmost consideration. In order to still have a secure algorithm, the designers have found the most suitable trade-off of security, area, and power at the area of 1570 GE and simulated power consumption of 5μW for PRESENT-80 and 1886 GE and 3.3μW for PRESENT-128 [6]. Its design was modeled after AES finalist Serpent and DES.

3.1 Structure of PRESENT

PRESENT is a Substitution Permutation Network(SPN) with 31 rounds. It takes its inputs as 64-bit blocks. Key size is 80 bit for PRESENT-80 and 128 bit for PRESENT-128 , hence the naming of the members. 80-bit key provides enough security for intended applications [6]. Similar to DES, in each round there is an XOR operation with the subkey, followed by a substitution and a permutation layer. The layers are explained below, with the bits with index zero representing the least significant and rightmost bit:

In **addRoundKey** step the round keys are extracted from the secret key as 64-bit

strings with a key schedule and are expressed as $K_i = \kappa^i_{63} \dots \kappa^i_0$ for $0 \leq i \leq 32$. Since the current STATE is in the form of $b_{63} \dots b_0$, the XOR operation this layer executes can be written as

$$b_j \rightarrow b_j \oplus \kappa^i_j \qquad \text{for} \qquad 0 \leq j \leq 63[6]$$

In **sBoxLayer** a 4-bit to 4-bit S-Box in $\mathbb{F}_2^4$ is used. The current STATE is divided into 16 4-bit words. In hexadecimal notation the values S-Box outputs for each input is given with a table.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 3.1: The S-Box lookup table of PRESENT [6]

This layer provides the non-linearity in the system and the above S-Box is used 16 times in parallel for every STATE.

In order for the S-Box to be resistant against differential and linear cryptanalysis attacks, the designers specify four conditions that the S-Box should satisfy. These conditions, taken directly from [6], are;

1. "For any fixed non-zero input difference $\Delta_I \in \mathbb{F}_2^4$ and any fixed non-zero output difference $\Delta_O \in \mathbb{F}_2^4$

$$\#\{x \in \mathbb{F}_2^4 | S(x) + S(x + \Delta_I) = \Delta_O\} \leq 4$$

2. For any fixed non-zero input difference $\Delta_I \in \mathbb{F}_2^4$ and any fixed output difference $\Delta_O \in \mathbb{F}_2^4$ such that $\text{wt}(\Delta_I) = \text{wt}(\Delta_O) = 1$

$$\{x \in \mathbb{F}_2^4 | S(x) + S(x + \Delta_I) = \Delta_O\} = \emptyset$$

3. For all non-zero $a \in \mathbb{F}_2^4$ and all non-zero $b \in \mathbb{F}_4$

$$|S_b^W(a)| \leq 8 \qquad \text{where} \qquad S_b^W(a) = \sum_{x \in \mathbb{F}_2^4} (-1)^{\langle b, S(x) \rangle + \langle a, x \rangle}$$

4. For all $a \in \mathbb{F}_2^4$ and all non-zero $b \in \mathbb{F}_4$ such that $\text{wt}(a) = \text{wt}(b) = 1$

$$S_b^W(a) = \pm 4 \qquad \text{where} \qquad S_b^W(a) \text{ is as above"}$$

The first condition is to make sure the algorithm is secure against differential cryptanalysis. If the S-Box satisfies this condition, that means all the values in the Difference Distribution Table is at most 4. Since the smaller these values the harder it is to break the algorithm using differential cryptanalysis, this is something the designers would desire.

If the S-Box satisfies the second condition, that would mean no input difference of 1 leads to an output difference of 1. Therefore, in any linear or differential path, any S-Box with one activated bit would activate at least two more S-Boxes in the next round. This is an important property in order to provide security against linear and differential cryptanalysis.

pLayer is the linear layer and in order to have efficient hardware implementations it uses a simple bit permutation. Just like the S-Box, this bit permutation can also be expressed with a table.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Table 3.2: The bit permutation of PRESENT [6]

The key schedule of PRESENT-80 and PRESENT-128 are very similar. The secret key is written onto the key register K. This key register is represented as $k_{79}k_{78}\ldots k_0$ for PRESENT-80 and as $k_{127}k_{126}\ldots k_0$ for PRESENT-128 . For each round, the key schedule first extracts the round key K_i, then updates the key register. For the extraction of the round key, 64 leftmost bits of the key register is taken [6].

$$K_i = \kappa_{63}\kappa_{62}\ldots\kappa_0 = k_{79}k_{78}\ldots k_{16} \qquad \text{for PRESENT-80}$$
$$K_i = \kappa_{63}\kappa_{62}\ldots\kappa_0 = k_{127}k_{126}\ldots k_{64} \qquad \text{for PRESENT-128}$$

The updating of the key register is slightly different for 80-bit key and 128-bit key. For 80-bit key, key register is rotated to the left for 61 bits, then the leftmost four bits

are replaced with their S-Box output. Finally five chosen bits are XORed with the `round_counter` to obtain the new key register [6]:

“

1. $[k_{79}k_{78}\dots k_0] = [k_{18}k_{17}\dots k_1k_0k_{79}k_{78}\dots k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \texttt{round\_counter}$

”

For 128-bit key, the updating of the key register also starts with a 61 bit left rotation. But this time, leftmost eight bits are replaced with their S-Box output, instead of four. Once again, chosen five bits are XORed with the `round_counter` and the key register is updated [6]:

“

1. $[k_{127}k_{126}\dots k_0] = [k_{66}k_{65}\dots k_1k_0k_{127}k_{126}\dots k_{67}]$
2. $[k_{127}k_{126}k_{125}k_{124}] = S[k_{127}k_{126}k_{125}k_{124}]$
3. $[k_{123}k_{122}k_{121}k_{120}] = S[k_{123}k_{122}k_{121}k_{120}]$
4. $[k_{66}k_{65}k_{64}k_{63}k_{62}] = [k_{66}k_{65}k_{64}k_{63}k_{62}] \oplus \texttt{round\_counter}$

”

3.2 Security Claims of PRESENT

The designers provide their security analysis in [6].

3.2.1 Differential Cryptanalysis

Theorem 3.2.1. *"Any five-round differential characteristic of* PRESENT *has a minimum of 10 active S-boxes [6]."*

The designers prove this theorem and use this as a base for their claim. According to the conditions the S-Box has to satisfy, the differential probability in one round can at most be 2^{-2}, and according to the theorem, in 25 rounds, $10 \times 5 = 50$ S-Boxes would be active at minimum. Therefore for 25 rounds, the differential probability would be $(2^{-2})^{50} = 2^{-100}$. It is possible, in differential cryptanalysis, to remove some rounds of the algorithm from the top, from the bottom, or from the both. However, even if an attacker somehow removes six rounds, the probability of success is very low.

3.2.2 Linear Cryptanalysis

Theorem 3.2.2. *"Let ϵ_{4R} be the maximal bias of a linear approximation of four rounds of* PRESENT *. Then $\epsilon_{4R} \leq \frac{1}{2^7}$ [6]."*

The designers prove and use another theorem for their claims. According to the theorem, for 28 rounds of PRESENT , the upper bound for the bias would be $2^6 \times (2^{-7})^7 = 2^{-43}$. Therefore, they state 2^{84} known plaintext/ciphertext pairs would be required to find the key [6].

3.2.3 Structural Attacks

The designers have also considered some structural attacks. However, structural attacks, they state, focus on words made up of bytes, but PRESENT is a bitwise algorithm [6]. Therefore they think PRESENT is secure against structural attacks.

3.2.4 Key Schedule Attacks

Key schedule attacks usually find some relations between different subkeys and exploit them. In order to provide security against key schedule attacks, the designers state that they used a round counter, and derived the subkeys from the original key in a non-linear way [6].

According to their analyses, the designers believe that PRESENT is a secure algorithm, but still encourage its further analysis.

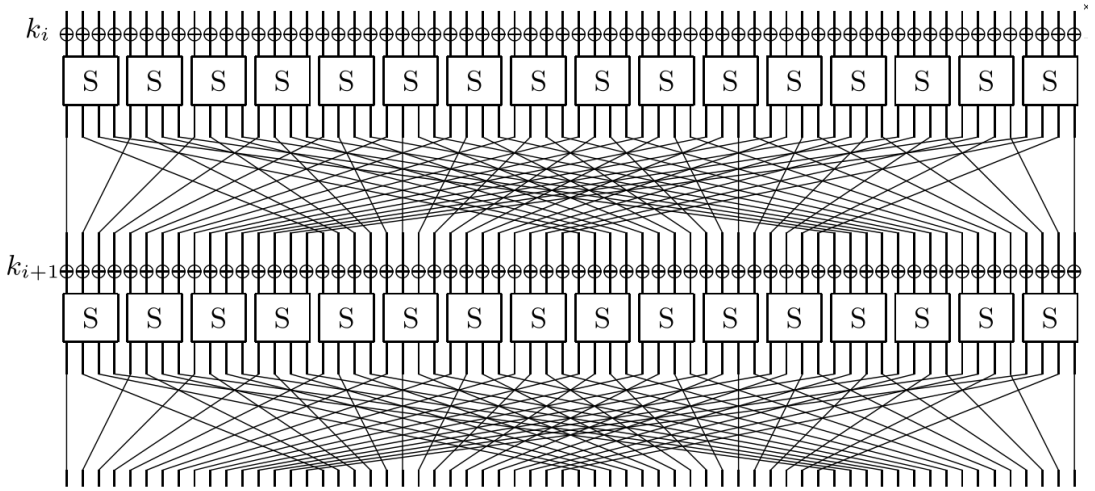


Figure 3.1: Two rounds of PRESENT [6]

CHAPTER 4

STREAM CIPHER TRIVIUM

TRIVIUM could be considered another good example of a lightweight algorithm. Like PRESENT , TRIVIUM is also a hardware-oriented algorithm. However, while PRESENT is a block cipher, TRIVIUM is a synchronous stream cipher. It was designed to see how much can a stream cipher be simplified without compromising its security and is a part of the eSTREAM project [8].

The designers state in [7] that as their main design consideration, they wanted the algorithm to generate keystream bits without linear correlations so that the keystream bits can't be exploited that way. In order to achieve that, they analyze some block ciphers and their linear characteristics, and apply them to stream ciphers by modifying them.

4.1 Structure of TRIVIUM

TRIVIUM is a stream cipher that uses an 80-bit secret key and an 80-bit initial value (IV) and can generate up to 2^{64} keystream bits. During the keystream generation process, it takes the internal state as 288 bits. In [8] the cipher is described in two phases, key stream generation and the initialization of the internal state using key and IV.

In **key stream generation** step, the internal state is first denoted as $(s_1, s_2, \ldots, s_{288})$. 6 bits are taken from this internal state and used to compute 1 bit of keystream. After that the internal state is updated by using 9 more bits from it. Application of these

steps iteratively, generates a keystream. The designers give a pseudocode of this process in their paper [8]:

“

for $i = 1$ to N **do**

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

”

The initialization of the internal state is done by loading the key and IV into the internal state. All the remaining bits of the internal state is initialized with a 0 bit, except for the last three bits, where they are initialized with a 1 bit. After that, the state is rotated for 4 cycles in the same way as the key stream generation process, without generating a key stream and the internal state is initialized. Once again, the designers give a pseudocode to explain the process [8]:

“

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

for $i = 1$ to $4 \cdot 288$ **do**

$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$

$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$

$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

”

In both psuedocodes $+$ and $\cdot$ operations represent XOR and AND respectively.

4.2 Security Claims of TRIVIUM

The designers mention in [8] a security requirement that they designed TRIVIUM to satisfy. This requirement is that any cryptographic attack should not be more easily implemented on TRIVIUM than on any other stream cipher with the same parameters. Then, they give some cryptographic properties of the cipher.

4.2.1 Correlations

There are two types of correlations that can be exploited when attacking a synchronous stream cipher. First is the correlations between the internal state bits and the key stream bits. It is easy to find this correlation in TRIVIUM , however, the state updates in a nonlinear way, therefore it should be difficult to exploit this correlation to recover the internal state [8].

The second type of correlation is between the key stream bits. To find a correlation of this type, linear trails can be followed like with the block ciphers and the number of AND gates activated can be counted. However, the designers state that they designed TRIVIUM to activate at least 72 AND gates in these types of linear trails [8]. Finding a correlation like this would require 2^{144} bits of the keystream. The designers also mention that other linear trails with bigger correlations might exist, but they wouldn't exceed 2^{-40} [8].

4.2.2 Period

The update of the state is done in a reversible way, and because of the way $(s_{178}, \ldots, s_{288})$ is initialized the state can only repeat itself after 111 iterations [8]. If TRIVIUM is assumed to act like a random permutation after some iterations, then all the cycles of length 2^{288} or less has the same probability of occuring [8]. Therefore for any key and IV, the probability that the period will be 2^{80} or less would be $\frac{2^{80}}{2^{288}} = 2^{-208}$ [8].

4.2.3 Guess and Determine Attacks

The designers mention that a straightforward attack would guess 195 bits in total, $(s_{25}, \ldots, s_{93})$, $(s_{97}, \ldots, s_{177})$, $(s_{244}, \ldots, s_{288})$ [8]. Finding the rest of the bits from the key stream would not be difficult. Therefore the designers recommend further research.

4.2.4 Algebraic Attacks

Because the state is updated in a nonlinear way, effective linearization techniques commonly used in algebraic attacks would be inefficient [8]. However, the designers remark that other techniques might be more efficient and encourage further analysis.

4.2.5 Resynchronization Attacks

Resynchronization attacks modify the IV, and tries to find the key by using the keystream [8]. But in TRIVIUM , after the key and IV is loaded into the state, the state is cycled 4 times. The designers believe this would counter the resynchronization attacks.

4.2.6 Cube Attacks

In 2009, Dinur and Shamir proposed Cube Attacks [9]. In this attack, different chosen IVs are analysed for the same key K, and used to find some linear equations for unknown bits in K. When enough number of linear equations are found, they can be

used to recover the secret key [4]. By using this attack, Dinur and Shamir attack the reduced round variants of TRIVIUM and while they were not able to break the full cipher, they have improved the previous known attacks. The best known attack for 672 initialization rounds was successfull with 2^{55} complexity, but with cube attack, this was reduced to 2^{19} [9]. For 735 and 767 initialization rounds no attack better than brute force existed, cube attacks can break these with 2^{30} and 2^{45} bit operations respectively [9]. Moreover, authors believe that with more resources, they can reduce the latter to about 2^{36} [9]. In 2016, Islam and Haq were able to recover 69 out of 80 bits of the key, for the reduced version of TRIVIUM of 576 rounds with $\mathcal{O}(2^{12.63})$ complexity [12].

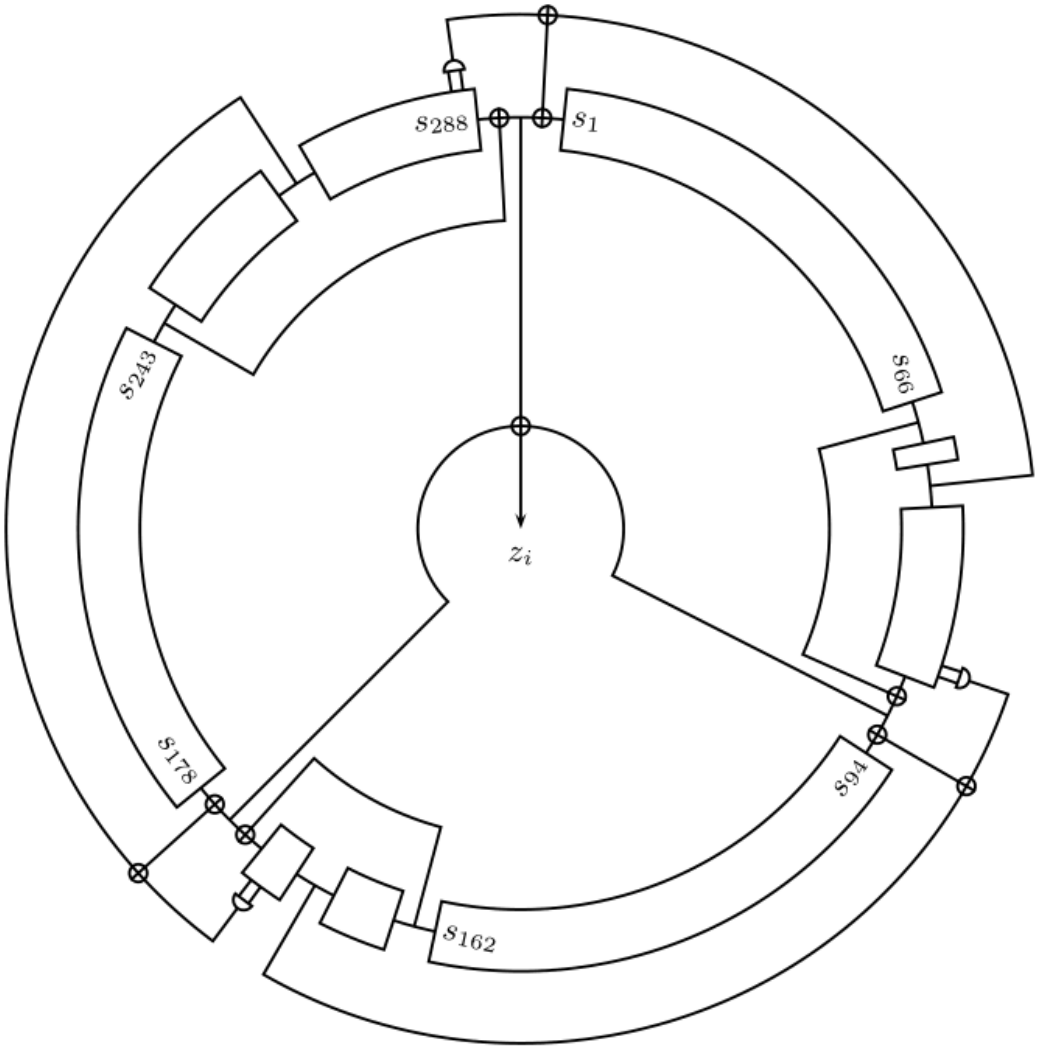


Figure 4.1: Key stream generation of TRIVIUM [8]

CHAPTER 5

LIGHTWEIGHT CRYPTOGRAPHY PROJECT

With the increased use of resource constrained devices, the need for efficient algorithms for these devices also increased. For this reason the National Institute of Standards and Technology (NIST) initiated a lightweight cryptography project in 2013 [15]. The goal was to study the current cryptographic algoritms and how they do in resource constrained environments, to see if lightweight algorithms are needed, and if they are, to design a standardization process. In order to achieve this goal, NIST held two workshops.

In late 2014, first workshop was announced and call for submissions was published [17]. In the call for submissions NIST describes the goal of the workshop, provides a list of topics for submissions, and gives specifications for them. In July 2015, the first Lightweight Cryptography Workshop was held in Gaithersburg, Maryland [19]. After the first workshop, in August 2016, NIST released a draft interagency report NISTIR 8114 and opened it for public comments [19]. In October 2016, second Lightweight Cryptography Workshop was held and shortly after that the public comment period for Draft NISTIR 8114 ended [19]. Then, in March 2017, NIST published the final draft of NISTIR 8114, *Report on Lightweight Cryptography* [15]. In this report, NIST first gives an overview of lightweight cryptography, including its target devices and performance metrics. After that, they outline the lightweight cryptography project, talking about its scope, design considerations, profiles, and evaluation process.

Scope of the project includes block ciphers, message authentication codes, hash ciphers, authenticated encryption schemes, stream ciphers and cryptographic permutations [15]. While public key cryptography is also mentioned to be in the scope of the

project, it is not included in the initial focus.

Design Considerations of the project includes security, flexibility, low overhead for multiple functions, ciphertext expansion, side channel and fault attacks, limits on the number of plaintext-ciphertext pairs, and related-key attacks [15]. It is explained that the security should be at least 112 bits. Algorithms that have similar encryption-decryption functions are preferred to algorithms that have different encryption and decryption functions. Ciphertext expansion is something that is not desired. Algorithms should be resistant against side channel and fault attacks. There may be an upper bound on the number of plaintext-ciphertext pairs. For some applications, resistance to related-key attacks may be desirable.

Some **profiles** would be established and NIST would evaluate algorithms according to the profiles. The characteristics that the profiles would address are given as follows [15]:

- Physical characteristics; area (GE, logic blocks, mm^2), memory (RAM and ROM), implementation type (hardware or software), energy (J)
- Performance characteristics; latency (clock cycles or time period), throughput (cycles per byte), power (W)
- Security characteristics; minimum security strength (bits), attack models, side channel resistance requirements

Lastly they talk about the **evaluation process**. NIST will hold workshops periodically to discuss the standardization process.

After the realese of NISTIR 8114, in April 2017 draft white paper *Profiles for the Lightweight Cryptography Standardization Process* was published and public comments were received until June 2017 [19]. This white paper was later retired. About a year later in May 2018 draft *Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process* was published along with a Federal Register Notice. In the Federal Register Notice, NIST requested comments on the proposed standardization process [2]. In August 2018, with another Federal Register Notice, the submission requirements were finalized and a call for lightweight

algorithm submissions was made [3]. Submission requirements also included the evaluation criteria. According to [18], the submitted algorithms would be evaluated on whether they meet the minimum acceptability requirements, side channel and fault attack resistance, cost, performance, third-party analysis, and suitability for hardware and software implementations. Minimum acceptability requirements include the security evaluation of the algorithm against known attacks.

The submission deadline was determined as late February 2019, with early January 2019 as early submission deadline [19]. If the algorithms were received by the early submission deadline, NIST would give feedback to them so they could be improved until the submission deadline. 57 algorithms in total were submitted for the project [25]. They were given until March 2019 to make amendments to their submissions and in April 2019, NIST announced the first round candidates [19]. 56 out of the 57 submissions were accepted. The full list of the first round candidates can be found in [20]. The duration of the first round was initially planned to be 12 months, but later was reduced to 4 [25]. In August 2019, 32 algorithms were announced as the second round candidates [19]. Later, in October 2019, NIST has published *NISTIR 8268, Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process* [25]. This report summarizes the lightweight cryptography project and highlights the acceptance and selection of the first and second round canditates. According to the report, selection of the second round canditates were made based on

- The maturity of the candidates: Candidates that did not have any third-party analysis and sufficent security claims were eliminated.
- Cryptanalysis of the candidates: Candidates that seemed insecure after the third-party analysis were eliminated.

In total 24 algorithms, 3 from maturity, 21 from cryptanalysis, were eliminated [25]. In the same report it is stated that performance will have more importance in selection of the third and final round candidates [25]. Full list of second round candidates with their classifications can be found in tables. Table 5.1 shows the algorithms that have both authenticated encryption with associated data(AEAD) and hashing functionalities and table 5.2 shows the algorithms with only the AEAD functionality.

Permutation Based
ACE
ASCON
DryGASCON
Gimli
KNOT
ORANGE
PHOTON-Beetle
SPARKLE (SCHWAEMM and ESCH)
Subterranean 2.0
Xoodyak
Block Cipher Based
Saturnin
Tweakable Block Cipher Based
SKINNY-AEAD/SKINNY-HASH

Table 5.1: Round 2 algorithms that provide both AEAD and hashing [25][21]

Permutation Based	Block Cipher Based
Elephant	COMET
ISAP	GIFT-COFB
Oribatida	HYENA
SPIX	mixFeed
SpoC	Pyjamask
WAGE	SAEAES
	SUNDAE-GIFT
	TinyJambu
Tweakable Block Cipher Based	**Stream Cipher Based**
ESTATE	Grain-128AEAD
ForkAE	
LOTUS-AEAD and LOCUS-AEAD	
Romulus	
Spook	

Table 5.2: Round 2 algorithms that provide only AEAD [25][21]

Most of the permutation based algorithms use a sponge construction. While some of the block cipher based algorithms uses their own block ciphers, the others use common block ciphers like GIFT-128 or AES. Tweakable block cipher based algorithms mostly use the tweakable block cipher `SKINNY`, or the tweakable versions of AES and GIFT, TweGIFT or TweAES. The only stream cipher based algorithm in round 2 is Grain-128AEAD, and it uses Grain family of stream ciphers.

The third and the last lightweight cryptography workshop as of now, was held in November 2019, at the NIST Main Campus Gaithersburg, Maryland. The accepted papers for the workshop include security analyses, security proofs, and benchmarking of the current and previous candidates. The next steps of the standardization process were also given. In [14] it is stated that third round selection is planned to be in September 2020, with the final selection in 2021. A fourth lightweight cryptography workshop is also planned. The chronology of NIST Lightweight Cryptography Project can be found in table 5.3.

Date	Event
July 2015	First Lightweight Cryptography Workshop
August 2016	Draft NISTIR 8114
October 2016	Second Lightweight Cryptography Workshop
October 2016	End of public comment period for draft NISTIR 8114
March 2017	NISTIR 8114, *Report on Lightweight Cryptography*
April 2017	Draft white paper *Profiles for the Lightweight Cryptography Standardization Process*
June 2017	End of public comment period for draft white paper
May 2018	Federal Register Notice
May 2018	Draft *Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process*
June 2018	End of public comment period for submission requirements
August 2018	Federal Register Notice
August 2018	*Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process*
January 2019	Early Submission Deadline
February 2019	Submission Deadline
March 2019	Amendment Deadline
April 2019	Announcement of the first round candidates
August 2019	Announcement of the second round candidates
October 2019	NISTIR 8268, *Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process*
November 2019	Third Lightweight Cryptography Workshop

Table 5.3: Timeline of Lightweight Cryptography Project [19]

In [23], benchmarking software implementations of round 1 candidates were given. The authors analyse the speed of the submission variants on four different microcontrollers and summarize their results. Five submissions and their variants from ten fastest implementations across the four microcontrollers made it to round 2 of the

lightweight cryptography project. These submissions are ESTATE, ASCON, KNOT, TinyJAMBU, and SAEAES. TinyJAMBU and GIMLI are the only two submissions from round 2 that have variants in ten smallest implementations. The authors also provide ten least RAM-intensive implementations on the STM32 F746ZG. Half of these implementations are variants of ASCON. KNOT and TinyJAMBU also have variants in these implementations. While RAM utilization measurements were only tested on one platform, the authors expect similar results on the other platforms [23].

In [24] six chosen algorithms from round 2, SpoC, GIFT-COFB, COMET-AES, COMET-CHAM, ASCON, and SCHWAEMM and ESCH, are implemented in different FPGAs. According to the implementation results, ASCON has the lowest latency for both AEAD and hashing. The authors compare the power and energy per bit of these chosen ciphers measured at 40 MHz on the Artix-7. As a result of this comparison, they found that the algorithms with the highest maximum frequencies were Spoc and ASCON-AEAD. ASCON-AEAD also has the highest throughput, while Spoc has the lowest area and power, and most energy efficient algorithms were found to be COMET-AES and ASCON-AEAD [24].

CHAPTER 6

ROUND 2 CANDIDATE ASCON

ASCON is one of the more well known algorithms in round 2 of the lightweight cryptography project. It was first published as part of the CAESAR competition. In the competition, the final portfolio was divided into three use cases, with the first use case being lightweight applications, and ASCON was the first choice for this use case [1]. During the competition, ASCON was analysed throughly. There are currently more than 15 publications analysing the security of ASCON and more than 80 analysing ASCON in a more general way [10]. All of these publications show the security of ASCON.

6.1 Structure of ASCON

ASCON uses authenticated ciphers ASCON-128 and ASCON-128a , and the hash function ASCON-HASH [10]. It operates on 320-bit states. The state is divided into two parts, rate r and capacity $c = 320 - r$. The exact number of r and c depends on the variant used [10].

For authenticated encryption, the designers identify two functions $\mathcal{E}$ and $\mathcal{D}$. $\mathcal{E}$ is the encryption function. It takes a secret key K, a nonce N, an associated data A, and a plaintext P, and returns a ciphertext C and a tag T.

$$\mathcal{E}_{k,r,a,b}(K, N, A, P) = (C, T)[10]$$

$\mathcal{D}$ is the decryption function. It takes the same secret key, nonce and associated data as the encryption function, along with the ciphertext C, and the tag T. If the tag is

verified it returns the plaintext P, otherwise it returns an error symbol $\perp$.

$$\mathcal{D}_{k,r,a,b}(K, N, A, C, T) = \{P, \perp\}[10]$$

In both of these equations k is the length of the key, r is the rate or the data block size, and a and b are the number of rounds. The parameter sets recommended by the authors are given in the table 6.1.

Name	Algorithms	Bit size of				Rounds	
		key	nonce	tag	data block	p^a	p^b
ASCON-128	$\mathcal{E}$, $\mathcal{D}_{128,64,12,6}$	128	128	128	64	12	6
ASCON-128a	$\mathcal{E}$, $\mathcal{D}_{128,128,12,8}$	128	128	128	128	12	8

Table 6.1: Recommended parameters for authenticated encryption [10]

For hashing an eXtendable output function is used. It takes a message M and returns a message digest H.

$$\mathcal{X}_{h,r,a}(M, \ell) = H[10]$$

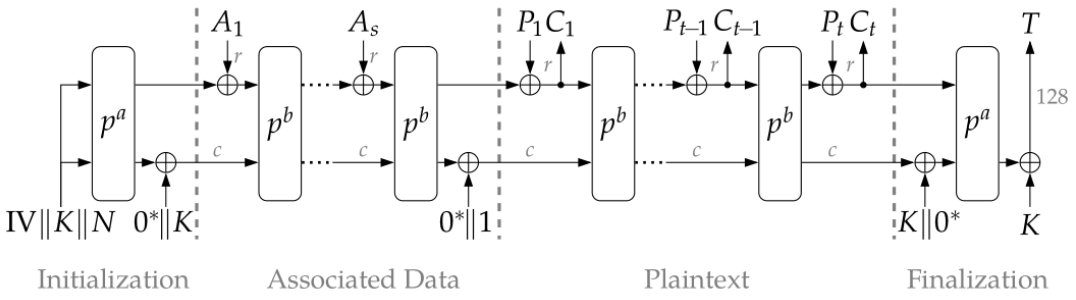
where h is the output length limit, r is the rate, a is the round number, and ℓ is the length of H. Once again, the designers give the recommended parameter set with a table.

Name	Algorithm	Bit size of		Rounds
		hash	data block	p^a
ASCON-HASH	$\mathcal{X}_{256,64,12}$ with $\ell = 256$	256	64	12

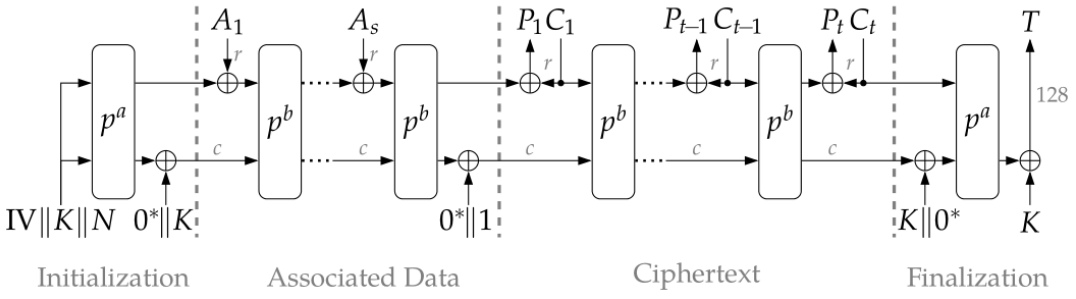
Table 6.2: Recommended parameters for hashing [10]

Authenticated encryption with ASCON starts with an initialization step. In this step an IV, the secret key K and the nonce N is concataneted to form the 320-bit state. The length of IV changes according to the variant of ASCON , but for ASCON-128 and ASCON-128a it is 64 bits. After that a transformation p is applied to the state for a times. Lastly the secrey key K is XORed with the state, and the initialization is completed. In the figure 6.1 0^* refers to a string of zeroes of arbitrary length.

After the initialization, the associated data is divided into r-bit blocks, and XORed with the first r bits of the state. After each XOR operation, transformation p is applied to the state b times. After the last block of associated data is XORed and the transformation is applied for one last time, the state is XORed with a 1-bit constant. This constant is called domain separation constant [10].



(a) Encryption $\mathcal{E}_{k,r,a,b}$



(b) Decryption $\mathcal{D}_{k,r,a,b}$

Figure 6.1: Authenticated encryption and decryption of ASCON [10]

After the associated data is incorporated into the system either an encryption or a decryption is done. For encryption, the plaintext is divided into r-bit blocks as in the processing of associated data. Each block of plaintext is then XORed with the first r bits of the state, and the ciphertext block is extracted. After each extraction of a ciphertext block, the state is updated with transformation p for b times. For decryption, the ciphertext blocks are first XORed with the first r bits of the state to extract the plaintext blocks, and then the ciphertext block replaces the first r bits of the state. After each usage of a ciphertext block, the state is once again updated with transformation p for b times.

The final step, after the encryption and decryption is done, is the finalization. In this step, the secret key K is XORed with the state. Then transformation p is applied to the state a times. The secret key K is once again XORed with the state. Least significant 128 bits are outputted as the tag T. The encryption function outputs the ciphertext blocks along with the T, while the decryption function only outputs the plaintext blocks if the tag is verified [10].

Hashing also starts with an initialization step. This time the state is initialized with

only an IV followed by necessary amount of zeroes. Transformation p is applied a times to end the initialization step. The message is divided into r-bit blocks. Each block is XORed with the first r bits of the state. After each XOR operation, the state is updated with transformation p for a times. After all the message blocks are used, the hash is extracted as r-bit blocks. After each extraction, tranformation p is applied to the state a times.

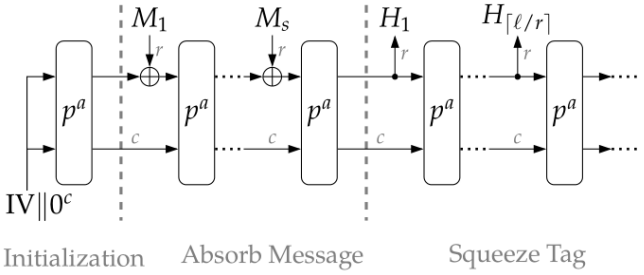


Figure 6.2: Hashing of ASCON [10]

Transformation p is a 320-bit permutation. The state is first divided into 5 words of length 64 bits.

$$S = x_0||x_1||x_2||x_3||x_4[10]$$

After that three layers, addition of constants (p_C), substitution layer (p_S), and linear diffusion layer (p_L) are applied in each round [10].

In **addition of constants**, round constants are added to the middle word x_2.

p^{12}	p^8	p^6	Round Constant	p^{12}	p^8	p^6	Round Constant
0			0000000000000000f0	6	2	0	0000000000000096
1			0000000000000000e1	7	3	1	0000000000000087
2			0000000000000000d2	8	4	2	0000000000000078
3			0000000000000000c3	9	5	3	0000000000000069
4	0		0000000000000000b4	10	6	4	000000000000005a
5	1		0000000000000000a5	11	7	5	000000000000004b

Table 6.3: Round constants of ASCON for 12, 8, and 6 rounds [10]

In **substitution layer** one bit from each word is taken in a bitslice fashion and they are replaced by using a 5-bit S-Box.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c
x	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$S[x]$	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17

Table 6.4: The S-Box lookup table of ASCON [10]

Linear diffusion layer mixes each word in itself to provide diffusion.

$$\text{“}x_0 \leftarrow x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)$$
$$x_1 \leftarrow x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)$$
$$x_2 \leftarrow x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)$$
$$x_3 \leftarrow x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)$$
$$x_4 \leftarrow x_4 \oplus (x_4 \ggg 7)1 \oplus (x_4 \ggg 41)[10]\text{”}$$

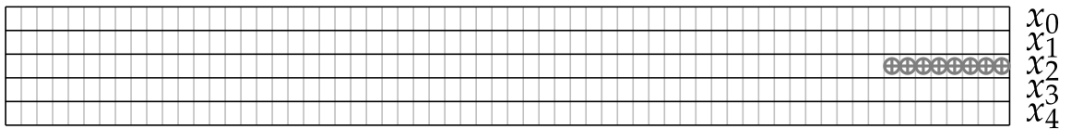


Figure 6.3: Round constant addition of ASCON [10]

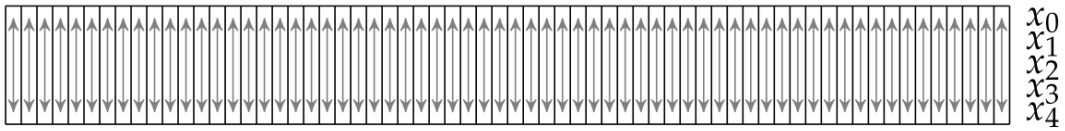


Figure 6.4: Substitution layer of ASCON [10]

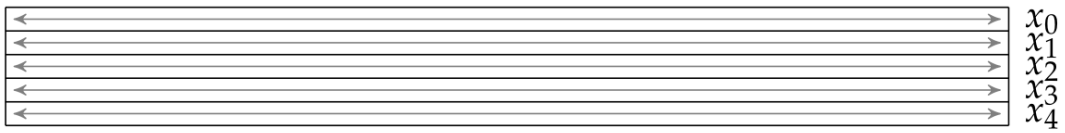


Figure 6.5: Linear diffusion layer of ASCON [10]

6.2 Security Analysis of ASCON

ASCON was first published as part of the CAESAR competition in 2014 [10]. Since then, it has been analysed by the cryptographic community, and the designers give an overview of the best attacks in their submission document. Best known key recovery attack can find the key with 2^{104} time complexity if the round number is reduced to 7 [10].

Type	Target	Rounds	Time
Key recovery	ASCON initialization	7/12	2^{104}
	ASCON initialization	5/12	2^{36}
Forgery	ASCON finalization	4/12	2^{101}
State recovery	ASCON-128a iteration	2/8	-

Table 6.5: Best known attacks on ASCON [10]

Type	Rounds	Time	Method
Distinguisher	12/12	2^{130}	Zero-sum
	11/12	2^{315}	Integral
	5/12	2^{193}	Differential
	5/12	2^{189}	Linear
Distinguisher	11/12	2^{85}	Zero-sum
	7/12	2^{65}	Integral
	5/12	2^{109}	Truncated Differential
	4/12	2^{107}	Differential
	4/12	2^{101}	Linear
Distinguisher	5/12	-	Zero-correlation
	5/12	-	Impossible Differential
	3/12	-	Subspace Trails

Table 6.6: Best known attacks on ASCON permutation [10]

The designers also mention that even if an attacker recovers the internal state at some point during the authenticated encryption, it is not feasible to use that to implement a key recovery or a forgery attack [10].

Some analysis of the permutation is also given. According to [10], in both linear approximation table (LAT), and the differential distribution table (DDT), the largest number is 2^{-2}, and its branch number is 3.

The designers analyse the S-Box as well. As per their findings, for more than 4 rounds, no linear and differential trail exist that use less than 64 active S-Boxes [10].

They were also able to find impossible differentials and zero-correlation linear hulls for 5 rounds, and while some impossible differentials for more than 5 rounds might exist, they don't think it can be used to implement any practical attack on ASCON [10].

The best known zero-sum distinguisher can be found with 2^{130} complexity for the full 12 rounds, or for 11 rounds with complexity 2^{85}, but the designers mention that no

known attack is able to exploit these properties to attack the authenticated encryption or hashing [10].

Best known attacks on the ASCON permutation is given with a table in [10] and the designers provide a list of published analysis of ASCON .

CHAPTER 7

CONCLUSION

With the recent technological changes, all kinds of devices, from powerful computing devices such as desktop computers, to small computing devices, such as RFID tags are being connected to each other via internet. With these changes, conventional cryptographic algorithms are slowly failing to satisfy the security and performance requirements especially on resource constrained devices. Therefore, the cryptographic community has been working to design efficient algorithms that can be implemented on resource constrained devices without compromising security or performance. PRESENT is one of these algorithms. It is a lightweight block cipher designed in 2007 [6], and since then, it has been analysed by the cryptographers. While there have been attacks on PRESENT , none of those attacks have been able to break the full 31-rounds of PRESENT . Thus, PRESENT remains as a good example of a lighweight algorithm. TRIVIUM can also be considered a good example in this context. It was designed in 2005 as part of the eSTREAM stream cipher project [7]. Like PRESENT , TRIVIUM has also been analysed since its submission. Because of its blockcipher-like design, and the nonlinear update of its state, it is secure against effective attacks on stream ciphers. Although both TRIVIUM and PRESENT seem secure algorithms, their designers strongly encourage further analysis.

With the increased focus on lightweight algorithms, NIST started a Lightweight Cryptography Project. The first goal of the project was to study the performance of current cryptographic algorithms on constrained devices and identify any need of lightweight standards [15]. To serve this purpose, NIST held two workshops. After these workshops, the standardization process has started. It was an open process, NIST received

public comments of any document it released. The project is still ongoing with 32 algorithms in round 2. Third round candidate selection is planned for September 2020 and the final results for 2021 [14].

REFERENCES

[1] *Cryptographic competitions CAESAR submissions*, 2019 (last accessed July 2020), Available at https://competitions.cr.yp.to/caesar-submissions.html.

[2] 83 Federal Register 22251 (May 14, 2018), pp 22251-22252, *Announcing Request for Comments on Lightweight Cryptography Requirements and Evaluation Criteria; Notice*, (last accessed July 2020), Available at https://www.federalregister.gov/documents/2018/05/14/2018-10127/announcing-request-for-comments-on-lightweight-cryptography-requirements-and-evaluation-criteria.

[3] 83 Federal Register 43656 (August 27, 2019), pp 43656-43657, *Announcing Request for Nominations for Lightweight Cryptographic Algorithms; Notice*, (last accessed July 2020), Available at https://www.federalregister.gov/documents/2018/08/27/2018-18433/announcing-request-for-nominations-for-lightweight-cryptographic-algorithms.

[4] S. Bedi and N. R. Pillai, Cube attacks on trivium., IACR Cryptol. ePrint Arch., 2009, p. 15, 2009.

[5] A. Biryukov and L. Perrin, State of the art in lightweight symmetric cryptography, 2017, Available at https://eprint.iacr.org/2017/511.pdf.

[6] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, Present: An ultra-lightweight block cipher, in *International workshop on cryptographic hardware and embedded systems*, pp. 450–466, Springer, 2007.

[7] C. De Cannière, Trivium: A stream cipher construction inspired by block cipher design principles, in *International Conference on Information Security*, pp. 171–186, Springer, 2006.

[8] C. De Cannière and B. Preneel, Trivium specifications, in *eSTREAM, ECRYPT Stream Cipher Project*, Citeseer, 2005.

[9] I. Dinur and A. Shamir, Cube attacks on tweakable black box polynomials, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 278–299, Springer, 2009.

[10] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, Ascon v1.2, Submission to NIST, 2019.

[11] M. Dworkin, Recommendation for block ciphers modes of operation methods and techniques, 2001, NIST Special Publication 800-38A.

[12] S. Islam and I. U. Haq, Cube attack on trivium and a5/1 stream ciphers, in *2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 409–415, IEEE, 2016.

[13] R. Kurzweil, The law of accelerating returns, in *Alan Turing: Life and Legacy of a Great Thinker*, pp. 381–416, Springer, Berlin, Heidelberg, 2004.

[14] K. McKay, *NIST Lightweight Cryptography Standardization: Next Steps*, National Institute of Standards and Technology, 2019, NIST Lightweight Cryptography Workshop 2019 Selected Presentations. Available at https://csrc.nist.gov/CSRC/media/Presentations/nist-lightweight-cryptography-standardization-next/images-media/session12-mckay-next-steps.pdf.

[15] K. A. McKay, L. Bassham, M. Sönmez Turan, and N. Mouha, Report on lightweight cryptography, 2017, NIST Internal Report (IR) 8114.

[16] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[17] National Institute of Standards and Technology, *Lightweight Cryptography Workshop 2015*, 2016 (last accessed July 2020), Available at https://www.nist.gov/news-events/events/2015/07/lightweight-cryptography-workshop-2015.

[18] National Institute of Standards and Technology, *Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process*, 2018 (last accessed July 2020), Available at https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf.

[19] National Institute of Standards and Technology, *Lightweight Cryptography Project Overview*, 2020 (last accessed July 2020), Available at https://csrc.nist.gov/Projects/lightweight-cryptography.

[20] National Institute of Standards and Technology, *Lightweight Cryptography Round 1 Candidates*, 2020 (last accessed July 2020), Available at https://csrc.nist.gov/projects/lightweight-cryptography/round-1-candidates.

[21] National Institute of Standards and Technology, *Lightweight Cryptography Round 2 Candidates*, 2020 (last accessed July 2020), Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[22] C. Paar and J. Pelzl, *Understanding Cryptography*, Springer, Berlin, Heidelberg, 2010.

[23] S. Renner, E. Pozzobon, and J. Mottok, Benchmarking software implementations of 1st round candidates of the NIST lwc project on microcontrollers, 2019.

[24] B. Rezvani, F. Coleman, S. Sachin, and W. Diehl, Hardware implementations of NIST lightweight cryptographic candidates: A first look., IACR Cryptol. ePrint Arch., 2019, p. 824, 2019.

[25] M. Sönmez Turan, K. A. McKay, Ç. Çalık, D. Chang, and L. Bassham, Status report on the first round of the NIST lightweight cryptography standardization process, 2019, NIST Internal Report (IR) 8268.

[26] D. R. Stinson, *Cryptography Theory and Practice*, Chapman & Hall/CRC, 2006.

[27] W. Trappe and L. C. Washington, *Introduction to Cryptography with Coding Theory*, Pearson Education, Inc., 2006.